Complex Systems **7** (1993) 389–413

Emerging Patterns and Computational Design of Filter Cellular Automata

A. S. Fokas
H. R. Madala
Department of Mathematics and Computer Science, Clarkson University, Potsdam, NY 13699, USA

Abstract. We study one-dimensional cellular automata using two different formulations, one polynomial and one involving mod 2. Emphasis is placed on the existence of coherent structures. Several new filter CAs are found that are capable of supporting either attractors or solitons. Some of these new rules are reversible.

1. Introduction

It is well established that even simple cellular automaton (CA) rules can exhibit a rich phenomenology. For example, Wolfram [1] considered the simple CA

$$a_i^{t+1} = \mathbf{f}(a_{i-2}^t + a_{i-1}^t + a_i^t + a_{i+1}^t + a_{i+2}^t) \tag{1.1}$$

where $a_i^t \in \{0, 1\}$ is the value of site i at time step t, and $\mathbf{f} \in \{0, 1\}$ is a function that specifies the CA rule. He found numerically that after a large number of iterations, one of four types of patterns could emerge from such a CA: i) a homogeneous state, ii) a set of separated simple stable or periodic structures, iii) an aperiodic (chaotic) pattern, and iv) a set of complex localized structures, sometimes long-lived. He also observed that the degree of regularity of the emerging patterns was related to the degree of irreversibility of the rules.

Park, Steiglitz, and Thurston [2] introduced a class of CAs called parity rule filter automata, and showed numerically that members of a certain subclass of these CAs exhibit solitonic behavior. An analytical study of such solitonic filter CAs was presented in [3–7]. Following these studies, several other types of soliton CAs have been introduced in the literature (see, for example, [8–11]).

A CA can be formulated in different ways; different formulations may be useful for studying different aspects of a CA. We consider two formulations, one polynomial, and one involving mod 2, in the following ways. We use a known solitonic CA to illustrate the formulations; we show how simple modifications of the existing solitonic CA yield CAs that also exhibit solitonic behavior; and we present an exhaustive numerical investigation of an important subclass of filter CAs, in order to identify those rules that are capable of supporting stable coherent structures. We distinguish between two cases: the soliton case, where the interaction properties of the coherent structures are not trivial; and the attractor case, where coherent structures either do not interact or fuse during interactions (Wolfram's class (ii) is of the latter type).

2. Mathematical formulation

A one-dimensional CA can be written in the form

$$a_i^{t+1} = \mathbf{f}\left(\sum_{j=r_1}^{r_2} a_{i+j}^t\right); \quad r_1, r_2 \in \mathcal{Z} \tag{2.1}$$

where $a_i^t \in \{0,1\}$ is the value of the site i at time step t, $\mathbf{f} \in \{0,1\}$ is an arbitrary function or rule, and the parameters r_1 and r_2 specify the local range of interactions. The CA given in (1.1) corresponds to $r_1 = -r_2 = -2$.

A one-dimensional filter CA can be written in the following form.

$$a_i^{t+1} = \mathbf{f}\left(\sum_{j=r_1}^{r_2} a_{i+j}^{t+1} + \sum_{j=r_2+1}^{r_3} a_{i+j}^t\right); \quad r_1, r_2, r_3 \in \mathcal{Z}, \quad r_1 \le r_2 < r_3 \tag{2.2}$$

In the case of the Park-Steiglitz-Thurston (PST) rule, $r_1 = -r, r_2 = -1, r_3 = r$, where $r \in \mathcal{Z}_+$, and $\mathbf{f}$ is the parity rule

$$\mathbf{f}(a) = \begin{cases} 0 & \text{if } a \text{ is odd or } 0 \\ 1 & \text{if } a \text{ is even} \end{cases} \tag{2.3}$$

Logical functions can always be reexpressed as ordinary polynomial functions. For example, the function $\mathbf{f}(v_1 + v_2 + v_3)$, where $\mathbf{f}, v_i \in \{0,1\}$, can be written as

$$\begin{aligned} \mathbf{f}(v_1 + v_2 + v_3) = \alpha_{123} v_1 v_2 v_3 + (\alpha_{12} v_1 v_2 \bar{v}_3 + \alpha_1 v_1 \bar{v}_2 \bar{v}_3 + CP) \\ + \alpha_0 \bar{v}_1 \bar{v}_2 \bar{v}_3 \end{aligned} \tag{2.4}$$

where $\bar{v} = (1 - v)$, CP denotes cyclic permutations, and $\alpha_{i_1 i_2 \ldots i_N} \in \{0,1\}$. Thus, there exist 2^{2^3} different rules depending on the choice of α. Similarly,

$$\begin{aligned} &\mathbf{f}(v_1 + v_2 + v_3 + v_4 + v_5) = \\ &\quad \alpha_{12345} v_1 v_2 v_3 v_4 v_5 + (\alpha_{1234} v_1 v_2 v_3 v_4 \bar{v}_5 + \alpha_{123} v_1 v_2 v_3 \bar{v}_4 \bar{v}_5 \\ &\quad + \alpha_{124} v_1 v_2 \bar{v}_3 v_4 \bar{v}_5 + \alpha_{12} v_1 v_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 + \alpha_{13} v_1 \bar{v}_2 v_3 \bar{v}_4 \bar{v}_5 \\ &\quad + \alpha_1 v_1 \bar{v}_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 + CP) + \alpha_0 \bar{v}_1 \bar{v}_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 \end{aligned} \tag{2.5}$$

and there exist 2^{2^5} such different rules.

It is evident that if $\mathbf{f}(a)$ is the parity rule defined in (2.3), then

$$\mathbf{f}(v_1 + \cdots + v_5) = v_1 v_2 v_3 v_4 \bar{v}_5 + v_1 v_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 + v_1 \bar{v}_2 v_3 \bar{v}_4 \bar{v}_5 + CP \tag{2.6}$$

that is, $\alpha_{1234} = \cdots = \alpha_{5123} = 1, \alpha_{12} = \cdots = \alpha_{51} = \alpha_{13} = \cdots = \alpha_{52} = 1$, and all other α in (2.5) equal 0.

It is sometimes convenient to use a mod 2 formulation (such a formulation was first given in [6]). For example, the parity rule (2.3) can be written as

$$\{v_1 + v_2 + v_3 + v_4 + v_5 + (1 - \bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4\bar{v}_5)\} \bmod 2 \tag{2.7}$$

Indeed, (2.6) implies

$$\mathbf{f}(v_1 + \cdots + v_5) = \Pi_2 - 3\Pi_3 + 7\Pi_4 - 15\Pi_5 \tag{2.8}$$

where Π_2 denotes all possible products between $v_1, \ldots, v_5$ taken two at a time, Π_3 denotes products taken three at a time, and so forth, with $\Pi_5 = v_1v_2v_3v_4v_5$. Also

$$\bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4\bar{v}_5 = 1 - \Pi_1 + \Pi_2 - \Pi_3 + \Pi_4 - \Pi_5 \tag{2.9}$$

Comparing (2.8) and (2.9), (2.7) follows.

The mod 2 formulation is particularly convenient for identifying reversible CAs (see [8, 9]).

Examples:

i) <u>The PST rule [2]</u>:

$$\begin{aligned} a_i^{t+1} &= \mathbf{f}(a_{i-2}^{t+1} + a_{i-1}^{t+1} + a_i^t + a_{i+1}^t + a_{i+2}^t); \\ \mathbf{f}(a) &= \begin{cases} 0 & \text{if } a \text{ is odd or } 0 \\ 1 & \text{if } a \text{ is even} \end{cases} \end{aligned} \tag{2.1}$$

admits the polynomial formulation

$$\begin{aligned} a_i^{t+1} = (&a_{i-2}^{t+1}a_{i-1}^{t+1}a_i^t a_{i+1}^t \bar{a}_{i+2}^t + a_{i-2}^{t+1}a_{i-1}^{t+1}\bar{a}_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t \\ &+ a_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}a_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t + CP) \end{aligned} \tag{2.2}$$

and the mod 2 formulation

$$\begin{aligned} a_i^{t+1} = \{&a_i^t + a_{i+1}^t + a_{i+2}^t - a_{i-1}^{t+1} - a_{i-2}^{t+1} \\ &+ (1 - \bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}\bar{a}_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t)\} \bmod 2 \end{aligned} \tag{2.3}$$

ii) <u>The reversible PST rule [9]</u>:

$$\begin{aligned} a_i^{t+1} &= \mathbf{f}(a_{i-2}^{t+1} + a_{i-1}^{t+1} + a_i^t + a_{i+1}^t + a_{i+2}^t); \\ \mathbf{f}(a) &= \begin{cases} 1 & \text{if } a \text{ is even or } a = a_i^t = 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.4}$$

admits the polynomial formulation

$$\begin{aligned} a_i^{t+1} = (&a_{i-2}^{t+1}a_{i-1}^{t+1}a_i^t a_{i+1}^t \bar{a}_{i+2}^t + a_{i-2}^{t+1}a_{i-1}^{t+1}\bar{a}_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t \\ &+ a_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}a_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t + CP) + \bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}a_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t \end{aligned} \tag{2.5}$$

and the mod 2 formulation

$$\begin{aligned} a_i^{t+1} &= \{a_i^t + a_{i+1}^t + a_{i+2}^t - a_{i-1}^{t+1} - a_{i-2}^{t+1} \\ &\quad + (1 - \bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}\bar{a}_{i+1}^t\bar{a}_{i+2}^t)\} \bmod 2 \end{aligned} \tag{2.6}$$

We note that the term $\bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}a_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t$ has been added to (2.11) and the term $\bar{a}_i^t$ has been deleted from (2.12). Formulations (2.12) and (2.15) indicate that while rule (2.10) is irreversible, rule (2.13) is reversible (see [9]).

iii) The JB rule [8]:

$$\begin{aligned} a_i^{t+1} &= \mathbf{f}(a_{i-2}^{t+1} + a_{i-1}^{t+1} + a_i^t + a_{i+1}^t + a_{i+2}^t + a_{i+3}^t); \\ \mathbf{f}(a) &= \begin{cases} 0 & \text{if } a = 0 \\ a_i^t & \text{if } a \text{ is odd} \\ \bar{a}_i^t & \text{if } a \text{ is even} \end{cases} \end{aligned} \tag{2.7}$$

The polynomial formulation considering $v_1 = a_i^t$, $v_2 = a_{i+1}^t$, $v_3 = a_{i+2}^t$, $v_4 = a_{i-1}^{t+1}$, $v_5 = a_{i-2}^{t+1}$, and $v_6 = a_{i+3}^t$ is

$$\begin{aligned} &\mathbf{f}(v_1 + v_2 + v_3 + v_4 + v_5 + v_6) = \\ &\quad v_1\bar{v}_2v_3v_4v_5v_6 + v_1v_2\bar{v}_3v_4v_5v_6 + v_1v_2v_3\bar{v}_4v_5v_6 + v_1v_2v_3v_4\bar{v}_5v_6 \\ &\quad + v_1v_2v_3v_4v_5\bar{v}_6 + \bar{v}_1\bar{v}_2v_3v_4v_5v_6 + \bar{v}_1v_2\bar{v}_3v_4v_5v_6 + \bar{v}_1v_2v_3\bar{v}_4v_5v_6 \\ &\quad + \bar{v}_1v_2v_3v_4\bar{v}_5v_6 + \bar{v}_1v_2v_3v_4v_5\bar{v}_6 + v_1\bar{v}_2\bar{v}_3\bar{v}_4v_5v_6 + v_1\bar{v}_2\bar{v}_3v_4\bar{v}_5v_6 \\ &\quad + v_1\bar{v}_2\bar{v}_3v_4v_5\bar{v}_6 + v_1\bar{v}_2v_3\bar{v}_4\bar{v}_5v_6 + v_1\bar{v}_2v_3\bar{v}_4v_5\bar{v}_6 + v_1\bar{v}_2v_3v_4\bar{v}_5\bar{v}_6 \\ &\quad + v_1v_2\bar{v}_3\bar{v}_4\bar{v}_5v_6 + v_1v_2\bar{v}_3\bar{v}_4v_5\bar{v}_6 + v_1v_2\bar{v}_3v_4\bar{v}_5\bar{v}_6 + v_1v_2v_3\bar{v}_4\bar{v}_5\bar{v}_6 \\ &\quad + \bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4v_5v_6 + \bar{v}_1\bar{v}_2\bar{v}_3v_4\bar{v}_5v_6 + \bar{v}_1\bar{v}_2\bar{v}_3v_4v_5\bar{v}_6 + \bar{v}_1\bar{v}_2v_3\bar{v}_4\bar{v}_5v_6 \\ &\quad + \bar{v}_1\bar{v}_2v_3\bar{v}_4v_5\bar{v}_6 + \bar{v}_1\bar{v}_2v_3v_4\bar{v}_5\bar{v}_6 + \bar{v}_1v_2\bar{v}_3\bar{v}_4\bar{v}_5v_6 + \bar{v}_1v_2\bar{v}_3\bar{v}_4v_5\bar{v}_6 \\ &\quad + \bar{v}_1v_2\bar{v}_3v_4\bar{v}_5\bar{v}_6 + \bar{v}_1v_2v_3\bar{v}_4\bar{v}_5\bar{v}_6 + v_1\bar{v}_2\bar{v}_3\bar{v}_4\bar{v}_5\bar{v}_6 \end{aligned} \tag{2.8}$$

and the mod 2 formulation is

$$\begin{aligned} a_i^{t+1} &= \{a_{i+1}^t + a_{i+2}^t + a_{i+3}^t - a_{i-1}^{t+1} - a_{i-2}^{t+1} \\ &\quad + (1 - \bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}\bar{a}_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t\bar{a}_{i+3}^t)\} \bmod 2 \end{aligned} \tag{2.9}$$

Again, the mod 2 formulation implies immediately that this rule is reversible. This rule is a special case of (2.2), with $r_1 = -2$, $r_2 = -1$, and $r_3 = 3$.

Other rules exhibiting solitonic structures are given in [10] and [11].

3. New solitonic rules

We present some new rules capable of supporting solitonic structures.

i) PST rule with weights:

$$\begin{aligned} a_i^{(t+1)} &= \mathbf{f}(a_{i-2}^{t+1} + 2a_{i-1}^{t+1} + 3a_i^t + 2a_{i+1}^t + a_{i+2}^t); \\ \mathbf{f}(a) &= \begin{cases} 0 & \text{if } a \text{ is odd or } 0 \\ 1 & \text{if } a \text{ is even} \end{cases} \end{aligned} \tag{3.1}$$

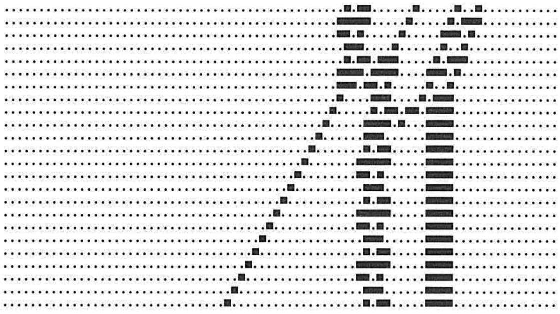

Figure 3.1: PST rule with weights (rule 1).
Coefficients: 0101101001011010101001011010010O.

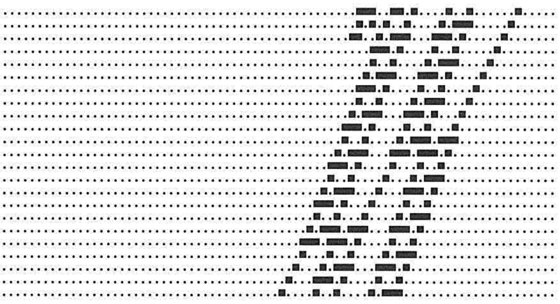

Figure 3.2: PST rule with weights (rule 1).
Coefficients: 01011010010110101010010110100100.

The polynomial formulation considering $v_1 = a_i^t$, $v_2 = a_{i+1}^t$, $v_3 = a_{i+2}^t$, $v_4 = a_{i-1}^{t+1}$, and $v_5 = a_{i-2}^{t+1}$ is given by

$$\begin{aligned}
&\mathbf{f}(3v_1 + 2v_2 + v_3 + 2v_4 + v_5) = \\
&\quad v_1v_2v_3v_4\bar{v}_5 + v_1v_2\bar{v}_3v_4v_5 + \bar{v}_1v_2v_3v_4v_5 + v_1v_2v_3\bar{v}_4\bar{v}_5 \\
&\quad + v_1v_2\bar{v}_3\bar{v}_4v_5 + v_1\bar{v}_2v_3v_4\bar{v}_5 + v_1\bar{v}_2\bar{v}_3v_4v_5 + \bar{v}_1v_2v_3\bar{v}_4v_5 \\
&\quad + \bar{v}_1\bar{v}_2v_3v_4v_5 + v_1\bar{v}_2v_3\bar{v}_4\bar{v}_5 + v_1\bar{v}_2\bar{v}_3\bar{v}_4v_5 + \bar{v}_1v_2\bar{v}_3v_4\bar{v}_5 \\
&\quad + \bar{v}_1\bar{v}_2v_3\bar{v}_4v_5 + \bar{v}_1v_2\bar{v}_3\bar{v}_4\bar{v}_5 + \bar{v}_1\bar{v}_2\bar{v}_3v_4\bar{v}_5
\end{aligned} \tag{3.2}$$

and the mod 2 formulation is

$$a_i^{t+1} = \{a_i^t + a_{i+2}^t - a_{i-2}^{t+1} + (1 - \bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}\bar{a}_i^t\bar{a}_{i+1}^t\bar{a}_{i+2}^t)\} \bmod 2 \tag{3.3}$$

Two typical solitonic patterns are shown in Figures 3.1 and 3.2.

ii) Reversible PST rule with weights:

$$a_i^{t+1} = \{a_i^t + a_{i+2}^t - a_{i-2}^{t+1} + (1 - \bar{a}_{i-2}^{t+1}\bar{a}_{i-1}^{t+1}\bar{a}_{i+1}^t\bar{a}_{i+2}^t)\} \bmod 2 \tag{3.4}$$

The polynomial formulation is the same as for the previous rule, with the additional term $v_1\bar{v}_2\bar{v}_3\bar{v}_4\bar{v}_5$. Typical solitonic patterns are shown in Figures 3.3 and 3.4.

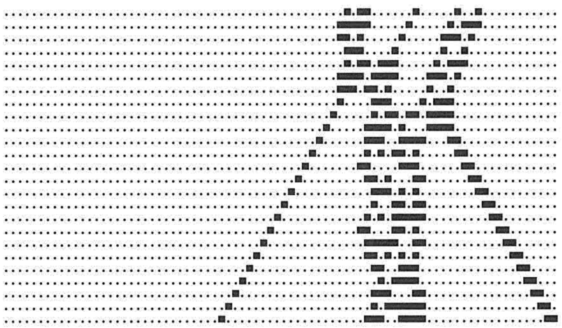
Figure 3.3: Reversible PST with weights (rule 2).
Coefficients: 01011010010110111010010110100100.

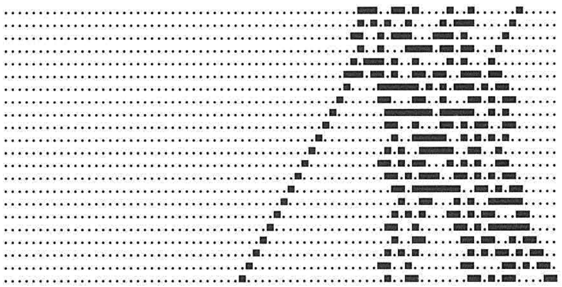
Figure 3.4: Reversible PST with weights (rule 2).
Coefficients: 01011010010110111010010110100100.

The extension of the two preceding rules to rules involving an arbitrary number of sites is straightforward.

iii) Other rules:

There exist several other cases of (2.2), with $r_1 = 2$, $r_2 = -1$, and $r_3 = 2$, that can support solitons. Some of these rules are summarized in Table 1, and typical solitonic patterns are shown in Figures 3.5–3.12. (Rules (3.1) and (3.4) appear in the table as rules 1 and 2, respectively.)

4. A systematic investigation of rules supporting coherent structures

Due to the large number of possible rules, it is practically impossible to investigate exhaustively all cases when the radius of interaction is 2. However, one can exhaustively study particular subclasses of CA and filter CA rules. These subclasses are specified by assuming that all α with the same number of indices are equal. In the case of (2.4), for example, we assume that $\alpha_{12} = \alpha_{13} = \alpha_{23}$ and $\alpha_1 = \alpha_2 = \alpha_3$. We study the subclasses for rules depending on 3 and on 5 sites.

Rule	α_{12345}	α_{1234}	α_{1235}	α_{123}	α_{1245}	α_{124}	α_{125}	α_{12}
1	0	1	0	1	1	0	1	0
2	0	1	0	1	1	0	1	0
3	0	0	0	1	1	0	0	1
4	0	0	1	1	0	1	1	0
5	0	1	1	0	0	0	0	0
6	0	1	1	0	1	1	1	1
7	1	1	1	1	1	0	0	0
8	0	0	0	0	1	0	0	1
9	1	1	0	1	1	0	0	1

Rule	α_{1345}	α_{134}	α_{135}	α_{13}	α_{145}	α_{14}	α_{15}	α_{1}
1	0	1	0	1	1	0	1	0
2	0	1	0	1	1	0	1	1
3	1	1	0	0	0	1	1	1
4	0	0	1	1	0	0	1	1
5	1	0	1	1	1	1	0	1
6	1	0	0	0	0	1	0	0
7	1	1	1	1	0	1	0	1
8	0	0	0	0	0	0	0	1
9	0	0	1	0	0	1	0	0

Rule	α_{2345}	α_{234}	α_{235}	α_{23}	α_{245}	α_{24}	α_{25}	α_{2}
1	1	0	1	0	0	1	0	1
2	1	0	1	0	0	1	0	1
3	1	0	1	0	1	0	1	1
4	0	1	0	1	0	0	0	0
5	1	0	1	0	1	0	0	1
6	0	0	0	1	0	1	0	0
7	1	0	0	0	0	1	0	1
8	0	0	1	1	1	1	0	0
9	0	0	0	0	1	0	1	1

Rule	α_{345}	α_{34}	α_{35}	α_{3}	α_{45}	α_{4}	α_{5}	α_{0}
1	1	0	1	0	0	1	0	0
2	1	0	1	0	0	1	0	0
3	0	0	0	0	0	0	0	0
4	0	1	0	0	1	0	0	0
5	1	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0
7	1	1	1	0	0	1	0	0
8	0	1	0	0	1	0	0	0
9	0	1	0	0	0	0	1	0

Table 1: CA rules with solitonic behavior.

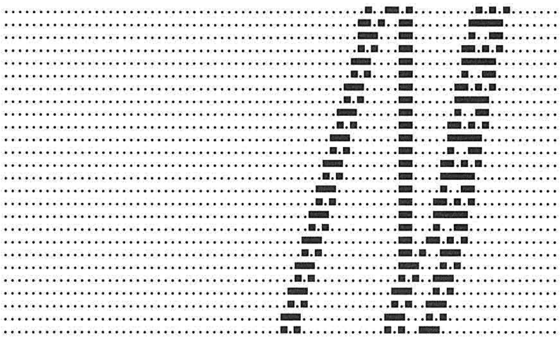
Figure 3.5: Rule 3.
Coefficients: 00011001110001111010101100000000.

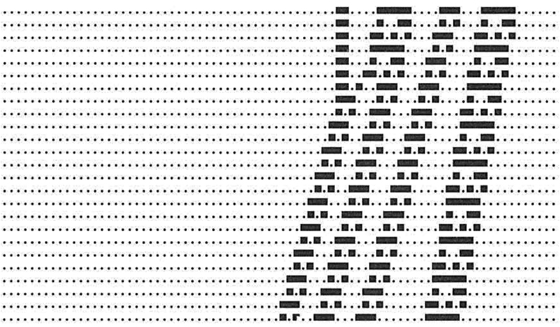
Figure 3.6: Rule 3.
Coefficients: 00011001110001111010101100000000.

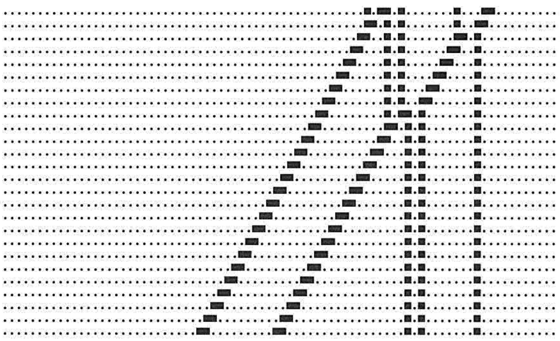
Figure 3.7: Rule 4.
Coefficients: 00110110001100110101000001001000.

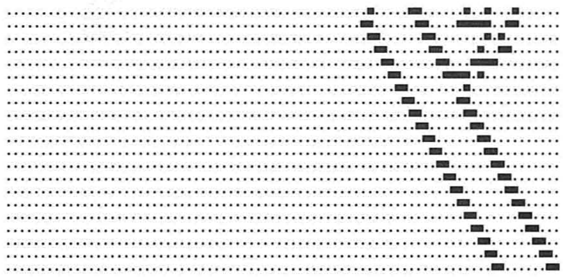
Figure 3.8: Rule 5.
Coefficients: 01100000101111011010100110000100.

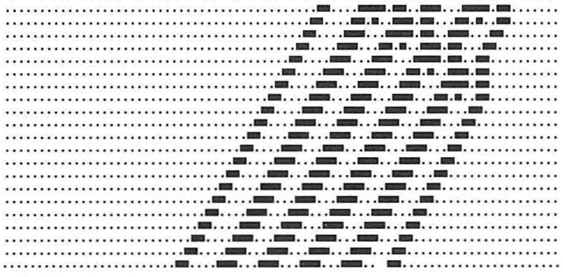
Figure 3.9: Rule 6.
Coefficients: 01101111100001000001010011000000.

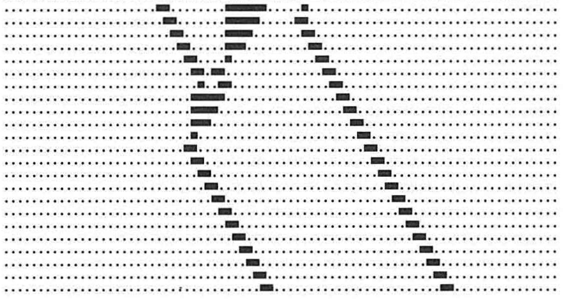
Figure 3.10: Rule 7.
Coefficients: 11111000111101011000010111100100.

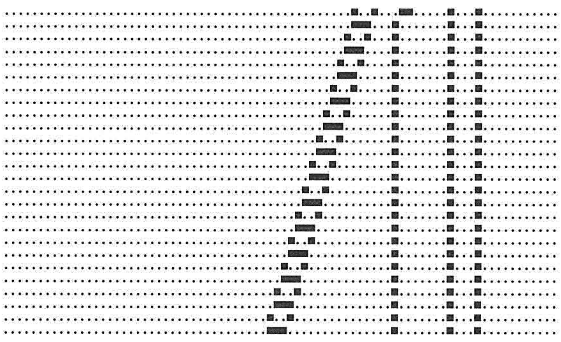

Figure 3.11: Rule 8.
Coefficients: 00001001000000010011110001001000.

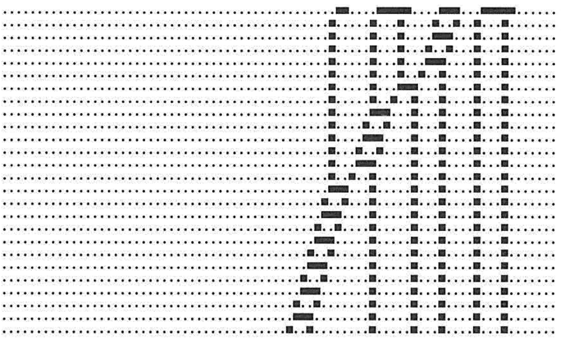

Figure 3.12: Rule 8.
Coefficients: 00001001000000010011110001001000.

I. Rules depending on 3 sites: We consider formulation (2.4) with

$$\begin{aligned} &\alpha_{123} = \alpha^{(3)}, \quad \alpha_{12} = \alpha_{13} = \alpha_{23} = \alpha^{(2)}, \\ &\alpha_1 = \alpha_2 = \alpha_3 = \alpha^{(1)}, \quad \text{and} \quad \alpha_0 = \alpha^{(0)} \end{aligned} \tag{4.1}$$

and consider first a particular CA and then a particular filter CA.

i)

$$a_i^{t+1} = \mathbf{f}(a_i^t + a_{i+1}^t + a_{i+2}^t) \tag{4.2}$$

In this case, we identify the following two rules that can support attractor structures.

$$\begin{array}{lccccccccc} & & \alpha_{123} & \alpha_{12} & \alpha_{13} & \alpha_1 & \alpha_{23} & \alpha_2 & \alpha_3 & \alpha_0 \\ \text{rule } 104 & \rightarrow & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ \text{rule } 232 & \rightarrow & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} \tag{4.3}$$

ii)

$$a_i^{t+1} = \mathbf{f}(a_{i-1}^{t+1} + a_i^t + a_{i+1}^t) \tag{4.4}$$

In this case, we identify the following four rules that can support attractors.

		α_{123}	α_{12}	α_{13}	α_1	α_{23}	α_2	α_3	α_0
rule 22	$\rightarrow$	0	0	0	1	0	1	1	0
rule 104	$\rightarrow$	0	1	1	0	1	0	0	0
rule 150	$\rightarrow$	1	0	0	1	0	1	1	0
rule 232	$\rightarrow$	1	1	1	0	1	0	0	0

(4.5)

Rule 104 is the PST rule for the radius $r = 1$.

II. <u>Rules depending on 5 sites:</u> We consider formulation (2.5) with

$$\alpha_{12345} = \alpha^{(5)}, \quad \alpha_{1234} = \cdots = \alpha_{2345} = \alpha^{(4)}, \quad \alpha_{123} = \cdots = \alpha_{345} = \alpha^{(3)},$$
$$\alpha_{12} = \cdots = \alpha_{45} = \alpha^{(2)}, \quad \alpha_1 = \cdots = \alpha_5 = \alpha^{(1)}, \quad \text{and} \quad \alpha_0 = \alpha^{(0)} \tag{4.6}$$

Again, we consider a particular CA, and a particular filter CA.

i)

$$a_i^{t+1} = \mathbf{f}(a_i^t + a_{i+1}^t + a_{i+2}^t + a_{i+3}^t + a_{i+4}^t) \tag{4.7}$$

In this case, we identify the following three rules that can support coherent structures.

		$\alpha^{(5)}$	$\alpha^{(4)}$	$\alpha^{(3)}$	$\alpha^{(2)}$	$\alpha^{(1)}$	$\alpha^{(0)}$
rule 20	$\rightarrow$	0	1	0	1	0	0
rule 24	$\rightarrow$	0	1	1	0	0	0
rule 56	$\rightarrow$	1	1	1	0	0	0

(4.8)

ii)

$$a_i^{t+1} = \mathbf{f}(a_{i-2}^{t+1} + a_{i-1}^{t+1} + a_i^t + a_{i+1}^t + a_{i+2}^t) \tag{4.9}$$

In this case, we identify the following thirteen rules that can support coherent structures. Rule 20 is the PST rule for the radius $r = 2$.

		$\alpha^{(5)}$	$\alpha^{(4)}$	$\alpha^{(3)}$	$\alpha^{(2)}$	$\alpha^{(1)}$	$\alpha^{(0)}$
rule 2	$\rightarrow$	0	0	0	0	1	0
rule 4	$\rightarrow$	0	0	0	1	0	0
rule 8	$\rightarrow$	0	0	1	0	0	0
rule 10	$\rightarrow$	0	0	1	0	1	0
rule 18	$\rightarrow$	0	1	0	0	1	0
rule 20	$\rightarrow$	0	1	0	1	0	0
rule 24	$\rightarrow$	0	1	1	0	0	0
rule 34	$\rightarrow$	1	0	0	0	1	0
rule 36	$\rightarrow$	1	0	0	1	0	0
rule 40	$\rightarrow$	1	0	1	0	0	0
rule 42	$\rightarrow$	1	0	1	0	1	0
rule 50	$\rightarrow$	1	1	0	0	1	0
rule 56	$\rightarrow$	1	1	1	0	0	0

(4.10)

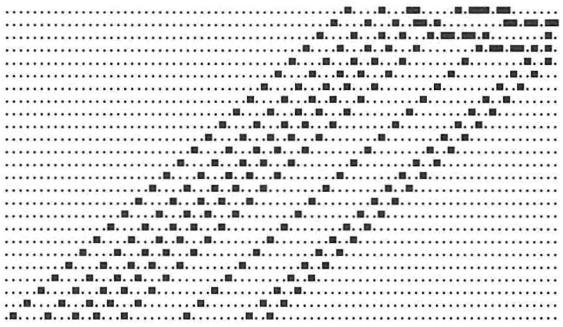
Figure 4.1: Rule 2. Coefficients: 0 0 0 0 1 0.

One can observe that if a rule supports coherent structure with coefficient $\alpha^{(5)} = 0$, then coherent structures also seem to be supported for the rule with $\alpha^{(5)} = 1$. This is true for rules 2 and 34, 4 and 36, 8 and 40, 10 and 42, 18 and 50, and 24 and 56. Rule 56 is the standard threshold automata, described as a majority rule in the neural modeling

$$\mathbf{f}(a) = \begin{cases} 1 & \text{if } a > \theta \\ 0 & \text{if } a \leq \theta \end{cases} \tag{4.11}$$

where θ is the threshold value of the unit; here it is equal to 2, as the range of the automata is $r = 2$. The polynomial formulation is represented in the case of CA as

$$\begin{aligned} a_i^{t+1} &= a_i^t a_{i+1}^t a_{i+2}^t a_{i+3}^t a_{i+4}^t + (a_i^t a_{i+1}^t a_{i+2}^t a_{i+3}^t \bar{a}_{i+4}^t \\ &\quad + a_i^t a_{i+1}^t a_{i+2}^t \bar{a}_{i+3}^t \bar{a}_{i+4}^t + a_i^t a_{i+1}^t \bar{a}_{i+2}^t a_{i+3}^t \bar{a}_{i+4}^t + CP) \end{aligned} \tag{4.12}$$

and in the case of filter CA as

$$\begin{aligned} a_i^{t+1} &= a_{i-2}^{t+1} a_{i-1}^{t+1} a_i^t a_{i+1}^t a_{i+2}^t + (\bar{a}_{i-2}^{t+1} a_{i-1}^{t+1} a_i^t a_{i+1}^t a_{i+2}^t \\ &\quad + \bar{a}_{i-2}^{t+1} \bar{a}_{i-1}^{t+1} a_i^t a_{i+1}^t a_{i+2}^t + \bar{a}_{i-2}^{t+1} a_{i-1}^{t+1} a_i^t a_{i+1}^t \bar{a}_{i+2}^t + CP) \end{aligned} \tag{4.13}$$

which is in a bounded form, in contrast to the unboundedness used in neural networks. This rule supports nonmovable patterns of attractor structures for the initial conditions of two particles.

Table 2 gives the types of pattern behavior for the preceding rules. Two types of attractor structures are observed, nonmovable and movable; nonmovable means that the initial data move straight down, and movable means that they move from right to left or left to right (see Figures 3.8 and 3.10). Some rules, such as 4 and 36 (Figures 4.2 and 4.9), support 'movable' patterns with a cyclic nature, indicating a periodic behavior.

An analytical investigation of these new rules that support coherent structures is beyond the scope of this paper. (We can only point out that there exists a Fast Rule Theorem [3] for the PST rule with weights that was introduced in section 3.) However, because we have followed the evolution of

Rule	Pattern behavior	Figure
2	attractors	(4.1)
4	attractors	(4.2)
8	nonmovable attractors	(4.3)
10	attractors	(4.4)
18	attractors	(4.5)
20	solitons	(4.6)
24	nonmovable attractors	(4.7)
34	attractors	(4.8)
36	attractors	(4.9)
40	nonmovable attractors	(4.10)
42	attractors	(4.11)
50	attractors	(4.12)
56	nonmovable attractors	(4.13)

Table 2: Specified rules and emerging pattern behaviors.

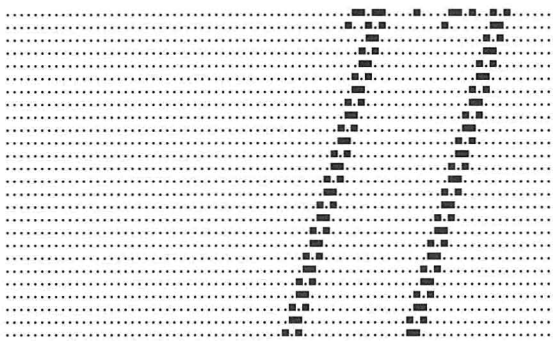
Figure 4.2: Rule 4. Coefficients: 0 0 0 1 0 0.

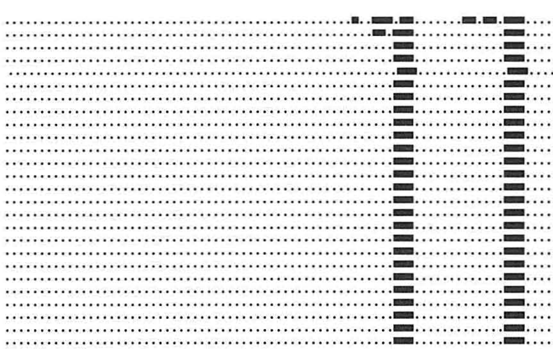
Figure 4.3: Rule 8. Coefficients: 0 0 1 0 0 0.

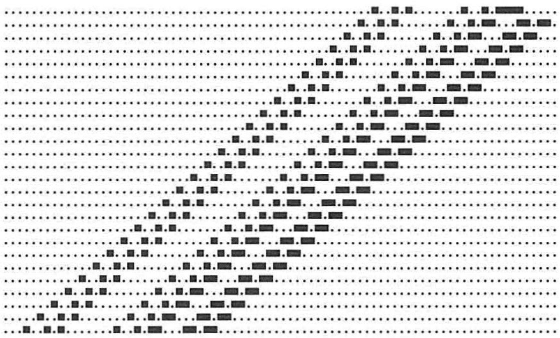
Figure 4.4: Rule 10. Coefficients: 0 0 1 0 1 0.

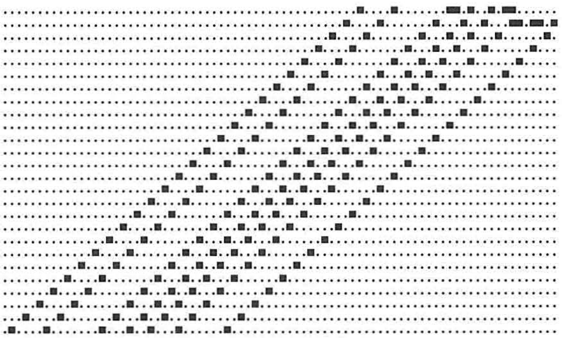
Figure 4.5: Rule 18. Coefficients: 0 1 0 0 1 0.

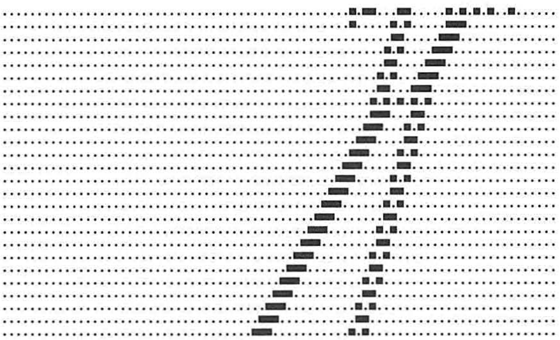
Figure 4.6: Rule 20. Coefficients: 0 1 0 1 0 0.

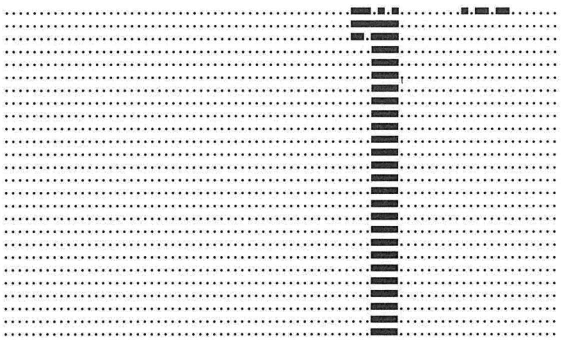
Figure 4.7: Rule 24. Coefficients: 0 1 1 0 0 0.

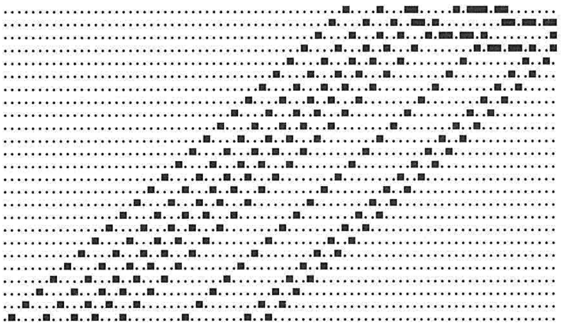
Figure 4.8: Rule 34. Coefficients: 1 0 0 0 1 0.

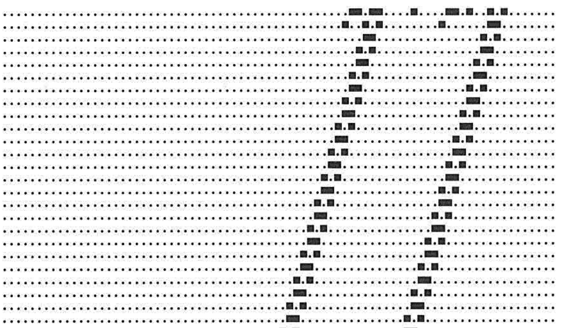
Figure 4.9: Rule 36. Coefficients: 1 0 0 1 0 0.

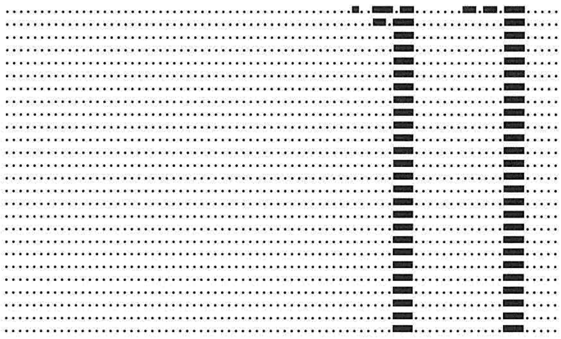
Figure 4.10: Rule 40. Coefficients: 1 0 1 0 0 0.

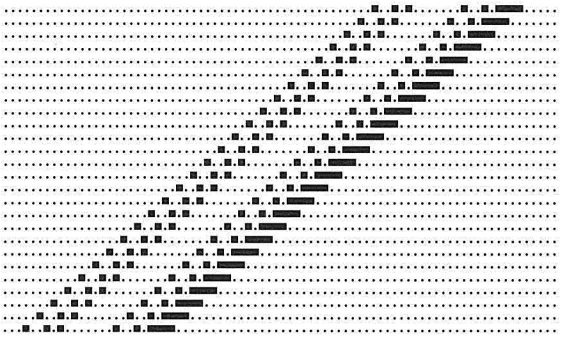
Figure 4.11: Rule 42. Coefficients: 1 0 1 0 1 0.

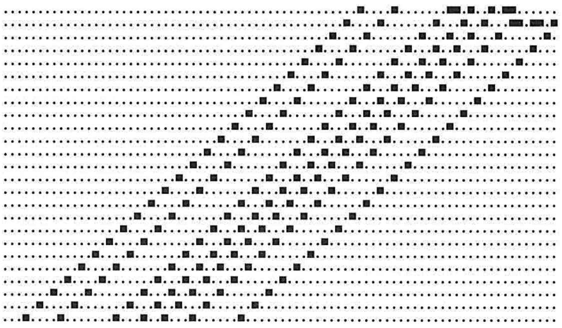
Figure 4.12: Rule 50. Coefficients: 1 1 0 0 1 0.

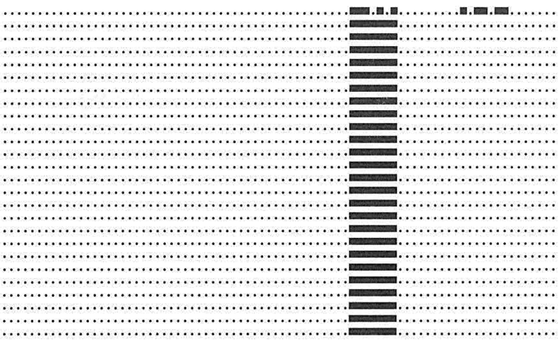

Figure 4.13: Rule 56. Coefficients: 1 1 1 0 0 0.

these rules starting with random initial data, we expect that the behavior we have observed is generic. In other words, for the rules given in Tables 1 and 2, we expect that any initial data will give rise to attractors, solitons, or a homogeneous state.

We note that there exist rules which for some initial data support solitons, which for other initial data yield chaotic patterns. Such a rule is (4.9) with $\alpha^{(5)} = \alpha^{(4)} = \alpha^{(2)} = 1, \alpha^{(3)} = \alpha^{(1)} = \alpha^{(0)} = 0$ (rule 52).

5. Conclusion

We have studied CA dynamics using both polynomial and mod 2 formulations. CAs usually support either attractor or chaotic pattern structures, whereas a large number of solitonic structures are possible with filter CAs. Figures 3.1–3.12 exhibit the solitonic behavior of various new CAs. Figures 4.1–4.13 exhibit the emerging coherent structures of other CA rules. Computational aspects of the algorithms are given for studying various CA rules by binary digit counters (Appendixes A and B). Our study shows that changing the polynomial formulations with unit weight connections results in various chaotic (unstable) and nonchaotic (stable) output patterns. This study can be extended to higher order polynomial formulations by slightly changing the given programs.

Appendix A: Generalized algorithm

In this appendix, we present a generalized algorithm for the CA and the filter CA found in (2.1) and (2.2), respectively. Let the radius $r = 1$; that is, there will be $2 * r + 1 (= 3)$ location sites considered for each state of formulation (2.4). In the case of the CA ($r_1 = 0$ and $r_2 = 2$), $v_1 = a_j^t$, $v_2 = a_{j+1}^t$, and $v_3 = a_{j+2}^t$ (forward locations) are taken, and in the case of the filter CA ($r_1 = -1$, $r_2 = -1$, and $r_3 = 1$), $v_1 = a_j^t$, $v_2 = a_{j+1}^t$, and $v_3 = a_{j-1}^{t+1}$ are taken. There are 2^3 terms, and the same number of coefficients in the polynomial formulation. Different rules are generated by giving values of 1s and 0s for

α_{123}, α_{12}, α_{13}, α_1, α_{23}, α_2, α_3, and α_0. Binary digit counters are used in the program. The algorithm generates two binary notation matrices; one for representing the terms in the polynomial equation, and another for the rules. A binary matrix of three is used to represent the terms, as shown in (A.1). Each row of the binary matrix represents one polynomial term in the equation.

$$\begin{array}{l}
000 \rightarrow \bar{v}_1\bar{v}_2\bar{v}_3 \rightarrow h(1) = \alpha_0 \\
001 \rightarrow \bar{v}_1\bar{v}_2 v_3 \rightarrow h(2) = \alpha_3 \\
010 \rightarrow \bar{v}_1 v_2\bar{v}_3 \rightarrow h(3) = \alpha_2 \\
011 \rightarrow \bar{v}_1 v_2 v_3 \rightarrow h(4) = \alpha_{23} \\
100 \rightarrow v_1\bar{v}_2\bar{v}_3 \rightarrow h(5) = \alpha_1 \\
101 \rightarrow v_1\bar{v}_2 v_3 \rightarrow h(6) = \alpha_{13} \\
110 \rightarrow v_1 v_2\bar{v}_3 \rightarrow h(7) = \alpha_{12} \\
111 \rightarrow v_1 v_2 v_3 \rightarrow h(8) = \alpha_{123}
\end{array} \tag{A.1}$$

where 1 represents v and 0 represents $\bar{v}(= 1 - v)$; and $h(i)$, $i = 1, \ldots, 8$ are the coefficient terms corresponding to each polynomial term and to α, as shown.

As there are eight coefficients, there are 2^8 (= 256) rules formed for all combinations of 0s and 1s. A binary matrix of eight is used to represent the rules (A.2).

$$\begin{array}{cccccccccl}
h(8) & h(7) & h(6) & h(5) & h(4) & h(3) & h(2) & h(1) & \leftarrow & \text{coefficient vector} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rightarrow & \text{rule } 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \rightarrow & \text{rule } 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \rightarrow & \text{rule } 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \rightarrow & \text{rule } 3 \\
 & & & & & & & & \vdots & \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \rightarrow & \text{rule } 255
\end{array} \tag{A.2}$$

Each row in the matrix represents a rule, and each element in it is one coefficient value. The rule numbers are computed as $h(8) * 2^7 + h(7) * 2^6 + h(6) * 2^5 + h(5) * 2^4 + h(4) * 2^3 + h(3) * 2^2 + h(2) * 2^1 + h(1) * 2^0$. As j varies from 1 to m, where m is the total number of state-space locations, the sites v_1, v_2, and v_3 are read for each state j, the polynomial terms are computed by using the binary counter notation (A.1) (if 1, the value of v_i is taken directly, and if 0, the value of v_i is subtracted from 1), and the state-space value of j at $t + 1$ is calculated.

Subroutines BMAT and BROW (see Appendix C) are used to generate the binary digit matrix. BROW generates the row of the matrix from the number of the row and the number of coefficients, using the parameters jq, jn, and $jd(jn)$, where jq is the variable to evaluate the specific row, jn is number of coefficients, and **jd** is the row vector of length jn. BMAT generates the binary matrix via BROW, using the parameters l, jf, and $id(jf, l)$, where l is number of coefficients or columns in the matrix, jf is total number of rows, and id is the binary matrix.

I. The first part of the algorithm is the initialization for the following values: m (number of spaces in the CA), t (number of time evolutions to be evolved), $a0(m)$ (initial conditions), ca (to choose the case as CA or filter CA), and r (to specify the radius of the CA).

```
c  PART A of Main Program
         character*1 u1(80)
         integer a0(80), a(80,2), v(50)
         integer h(50), h1(50), ide(512,50)
         integer r,t,s1,ca,o2
         real ic
c  Initialization
 110     format('CA case (1) / FilterCA case (2) ??')
 120     format('Give r value ???')
         m =80
         t =25
         seed =0.0
           write(*,110)
           read(*,*)ca
           write(*,120)
           read(*,*)r
           s1 =2*r+1
           if (ca.eq.1) r =s1-1
         do 1 j = 1,m
  1      a0(j) = 0
         do 2 i =m-13-r,m-r-3
          k =i-15
          a0(k) =RND(seed)
  2      a0(i) =RND(seed)
```

In this example, m and t have been set at 80 and 25, respectively. Initial input data of $a0(j); j = 1, \ldots, m$ are generated by using a random function with an initial seed value set at 0 (a pseudorandom generating function is given in Appendix C).

The values of ca and r are to be fed as inputs to the program. The CA case requires an input of $1\,(= ca)$ and the filter CA case requires an input of $2\,(= ca)$. In the program, the value of $s1$ is computed as $s1 = 2 * r + 1$, for obtaining the total number of arguments v.

II. The binary matrix for $s1$ is generated with 2^{s1} rows and $s1$ columns as in (A.1); BMAT generates the matrix ide with $kf (= 2^{s1})$ rows.

III. The coefficient vector $\mathbf{h}$ as a CA rule (A.2) is generated using BROW, according to the number of terms kf in the polynomial. In part B of the main program, BMAT and BROW are used, and initial conditions $a0(80)$ are transferred to $a(i, 1)$ as $t = 1$. The rule number $kf2$ is given as $h1(1) * 2^{s1-1} + h1(2) * 2^{s1-2} + \cdots + h1(s1) * 2^0$.

```
c  PART B of Main Program
c   Generation of binary matrix
 100      format(80a1)
 130      format('Rule #',i3,1x,'Coeffts:',33i2)
          call BMAT(s1,kf,ide)
            ic =0.
  11      call BROW(ic,kf,h1)
            kf2 =0
             do 3 i = 1,kf
              i1 =kf-i+1
              h(i1) =h1(i)
   3         kf2 =kf2+h1(i)*2**(i1-1)
            ic =ic+1
          do 4 i = 1,m
           a(i,1) =a0(i)
           a(i,2) =0
            if (a(i,1).eq.0) u1(i) =' '
            if (a(i,1).eq.1) u1(i) ='1'
   4      continue
             write(*,130)kf2, (h1(i),i=1,kf)
             write(*,100) (u1(j),j=1,m)
```

IV. The next step is to compute the time evolution at time $t + 1$, and to proceed to obtain an evolutionary pattern of the CA. At each state j, the location sites $(v_1, v_2, \ldots, v_{s1})$ are formed, and $a(j, 2)$ is computed at $t+1 = 2$. Before going to the next time step, $a(j, 2)$, j =1 to m is printed and the values are transferred to $a(j, 1)$. This procedure is repeated until all the specified time steps t are performed.

```
c  PART C of Main Program
          do 9 k = 1,t
            do 7 j = 1,m
c   forming vector v as per CA or filterCA case
              i =0
              do while (i.le.r)
                v(i+1) =0
                if (j+i.le.m) v(i+1) =a(j+i,1)
                if (ca.eq.2.and.i.ne.0) then
                  v(i+r+1) =0
                  if (j-i.ge.1) v(i+r+1) =a(j-i,2)
                endif
                i =i+1
              enddo
c   evolution at next time step
              a(j,2) =0
              do 6 li = 1,kf
```

```
                o2 =1
                do 5 lj = 1,s1
                  if (ide(li,lj).eq.0) then
                    o2 =o2*(1-v(lj))
                   else
                    o2 =o2*v(lj)
                  endif
      5         continue
                a(j,2) =a(j,2)+h(li)*o2
      6       continue
      7     continue
            do 8 j = 1,m
              a(j,1) = a(j,2)
              a(j,2) = 0
                  if (a(j,1).eq.0) u1(j) =' '
                  if (a(j,1).eq.1) u1(j) ='1'
      8     continue
            write(*,100)(u1(j),j =1,m)
      9   continue
c     stopping after all the rules are covered
            is =0
          do 10 j1=1,kf
     10   is=is+h(j1)
          if (is.lt.kf) goto 11
        stop
        end
```

V. This process is repeated for all 2^{s1} rules, using the same initial conditions. It is possible to choose the coefficient vector for a specific CA rule as an input each time.

Appendix B: Algorithm for specified rules

The generalized algorithm given in Appendix A is simplified here to study specified rules of CA and filter CA cases. This is preferable for the study of CAs under specific conditions as given, and reduces the computational burden when $r \geq 2$.

For example, let us consider $r = 2$, thus $s1 = 2 * r + 1 (= 5)$ location sites are considered at each state of the CA. The polynomial formulation (2.5) is specified as follows.

$$\begin{aligned} &\mathbf{f}(v_1 + v_2 + v_3 + v_4 + v_5) = \\ &\quad \alpha^{(5)} v_1 v_2 v_3 v_4 v_5 + \alpha^{(4)} (v_1 v_2 v_3 v_4 \bar{v}_5 + CP) + \\ &\quad \alpha^{(3)} (v_1 v_2 v_3 \bar{v}_4 \bar{v}_5 + v_1 v_2 \bar{v}_3 v_4 \bar{v}_5 + CP) + \\ &\quad \alpha^{(2)} (v_1 v_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 + v_1 \bar{v}_2 v_3 \bar{v}_4 \bar{v}_5 + CP) + \\ &\quad \alpha^{(1)} (v_1 \bar{v}_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 + CP) + \alpha^{(0)} \bar{v}_1 \bar{v}_2 \bar{v}_3 \bar{v}_4 \bar{v}_5 \end{aligned} \tag{B.1}$$

where v_i are the location sites considered, $\bar{v} = (1 - v)$, and $\alpha^{(i)} \in \{0, 1\}$ are the coefficients. In the CA case ($r_1 = 0$ and $r_2 = 4$), we take $v_1 = a_j^t, v_2 = a_{j+1}^t, v_3 = a_{j+2}^t, v_4 = a_{j+3}^t$, and $v_5 = a_{j+4}^t$; and in the filter CA case ($r_1 = -2, r_2 = -1$, and $r_3 = 2$), we take $v_1 = a_j^t, v_2 = a_{j+1}^t, v_3 = a_{j+2}^t, v_4 = a_{j-1}^{t+1}$, and $v_5 = a_{j-2}^{t+1}$. As each polynomial term is formed with five site terms, a binary matrix of five is used. This is given by $ide(i, j); j = 1, \ldots, s1, i = 1, \ldots, kf$ where $s1\,(= 5)$ is the number of columns and $kf (= 2^5)$ is the number of rows, which represents the number of polynomial terms ($= 32$). As previously, each row of the matrix corresponds to one term in the polynomial.

$$\begin{array}{l}
00000 \to \bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4\bar{v}_5 \to h(0) = \alpha^{(0)} \\
00001 \to \bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4 v_5 \to h(1) = \alpha^{(1)} \\
00010 \to \bar{v}_1\bar{v}_2\bar{v}_3 v_4\bar{v}_5 \to h(1) = \alpha^{(1)} \\
00011 \to \bar{v}_1\bar{v}_2\bar{v}_3 v_4 v_5 \to h(2) = \alpha^{(2)} \\
00100 \to \bar{v}_1\bar{v}_2 v_3\bar{v}_4\bar{v}_5 \to h(1) = \alpha^{(1)} \\
00101 \to \bar{v}_1\bar{v}_2 v_3\bar{v}_4 v_5 \to h(2) = \alpha^{(2)} \\
00110 \to \bar{v}_1\bar{v}_2 v_3 v_4\bar{v}_5 \to h(2) = \alpha^{(2)} \\
00111 \to \bar{v}_1\bar{v}_2 v_3 v_4 v_5 \to h(3) = \alpha^{(3)} \\
\quad\vdots \qquad\qquad\quad \vdots \qquad\quad \vdots \\
11110 \to v_1 v_2 v_3 v_4\bar{v}_5 \to h(4) = \alpha^{(4)} \\
11111 \to v_1 v_2 v_3 v_4 v_5 \to h(5) = \alpha^{(5)}
\end{array} \tag{B.2}$$

where the terms $h(0) = \alpha^{(0)}$, $h(1) = \alpha^{(1)}$, $h(2) = \alpha^{(2)}$, $h(3) = \alpha^{(3)}$, $h(4) = \alpha^{(4)}$, and $h(5) = \alpha^{(5)}$ are specified according to the number of 1s in the binary term. As there are six coefficients, there are 2^6 ($= 64$) rules. We consider only those rules where $h(0) = 0$. This enables us to use the same binary matrix (B.1) for indicating the rules.

$$\begin{array}{cccccccl}
h(5) & h(4) & h(3) & h(2) & h(1) & h(0) & \leftarrow & \text{coefficient vector} \\
 & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & \to & \text{rule } 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \to & \text{rule } 2 \\
0 & 0 & 0 & 1 & 0 & 0 & \to & \text{rule } 4 \\
0 & 0 & 0 & 1 & 1 & 0 & \to & \text{rule } 6 \\
 & & & & & & \vdots & \\
1 & 1 & 1 & 1 & 1 & 0 & \to & \text{rule } 62
\end{array} \tag{B.3}$$

Consequently, each row of the matrix *ide* is also considered as the coefficient terms $h(i), i = 0, 1, \ldots, s1$, by including the term $h(0)$ as $\alpha^{(0)}$ (B.3); this is done by extending the columns of the matrix *ide* to $s1 + 1$. Each coefficient vector becomes a rule, and each term of the vector becomes the connecting coefficient value of the polynomial term, according to the total number of 1s in the binary counter of that term. The specified rule numbers are computed as $h(5) * 2^5 + h(4) * 2^4 + h(3) * 2^3 + h(2) * 2^2 + h(1) * 2^1 + h(0) * 2^0$. The program given in Appendix A is here simplified, and it works for the specified rules having $h(0) = 0$.

```
c  PROGRAM  for specified rules
c   PART A
          character*1 u1(80)
          integer a0(80), a(80,2), v(50)
          integer h(50), ide(512,50)
          integer r,t,s1,ca,o2
  100     format(80a1)
  110     format('CA case (1) / FilterCA case (2) ??')
  120     format('Give r value ???')
  130     format('Rule #',i3,1x,'Coeffts:',30i2)
          m =80
          t =25
          seed =0.0
            write(*,110)
            read(*,*)ca
            write(*,120)
            read(*,*)r
            s1 =2*r+1
            if (ca.eq.1) r =s1-1
          do 1 j = 1,m
    1     a0(j) = 0
          do 2 i =m-13-r,m-r-3
           k =i-15
           a0(k) =RND(seed)
    2     a0(i) =RND(seed)
c   PART B
          call BMAT(s1,kf,ide)
          do 10 ic = 1,kf
            kf2 =0
            ide(ic,s1+1) =0
            do 3 i1 = 1,s1+1
              i =s1-i1+1
              h(i) =ide(ic,i1)
    3       kf2 =kf2+ide(ic,i1)*2**i
          do 4 i = 1,m
           a(i,1) =a0(i)
           a(i,2) =0
            if (a(i,1).eq.0) u1(i) =' '
            if (a(i,1).eq.1) u1(i) ='1'
    4     continue
             write(*,130)kf2, (ide(ic,i),i=1,s1+1)
             write(*,100) (u1(j),j=1,m)
c   PART C
          do 9 k = 1,t
            do 7 j = 1,m
              i =0
```

```
               do while (i.le.r)
                 v(i+1) =0
                 if (j+i.le.m) v(i+1) =a(j+i,1)
                 if (ca.eq.2.and.i.ne.0) then
                   v(i+r+1) =0
                   if (j-i.ge.1) v(i+r+1) =a(j-i,2)
                 endif
                 i =i+1
               enddo
               a(j,2) =0
               do 6 li = 1,kf
                 kx =0
                 o2 =1
                 do 5 lj = 1,s1
                   if (ide(li,lj).eq.0) then
                     o2 =o2*(1-v(lj))
                    else
                     kx =kx+1
                     o2 =o2*v(lj)
                   endif
 5               continue
                   a(j,2) =a(j,2)+h(kx)*o2
 6             continue
 7           continue
             do 8 j = 1,m
               a(j,1) = a(j,2)
               a(j,2) = 0
                   if (a(j,1).eq.0) u1(j) =' '
                   if (a(j,1).eq.1) u1(j) ='1'
 8           continue
             write(*,100)(u1(j),j =1,m)
 9         continue
10         continue
         stop
         end
```

Appendix C: Subroutines used

```
   subroutine BMAT(l,jf,id)
   dimension ip(50),id(512,50)
   real ic
     ic =0.0
     ict =1
     jf =0
25   call BROW(ic,l,ip)
       ic =ic+1
```

```
   subroutine BROW(jq,jn,jd)
   dimension jd(50)
   real jq, jl
    js =jn+1
10  jl =jq
    do 11 i = 1,jn
11  jd(i) =0
    if (js-1) 15,21,15
```

```
        is =0
       do 26 j1 = 1,l
26     is =is+ip(j1)
       jf =jf+1
       do 27 j = 1,l
27     id(jf,j) =ip(j)
       write(*,28)(id(jf,j),j =1,l)
28      format(5x,17i2)
       if(is-ict) 25,30,30
30   return
      end

c  Random number generating function

     function RND(seed)
      x1 =(seed+3.14159)*5.04
      x1 =x1-int(x1)
       seed =x1
      if (x1.gt.0.5) RND =1
      if (x1.le.0.5) RND =0
     return
      end
```

```
15     i =0
       jn1 =jn+1
16     i =i+1
       if (js-jl) 17,17,18
17     jc =jl/js
       jd(jn1-i) =jl-jc*js
       jl =jc
       goto 16
18     jd(jn1-i) =jl
       do 20 i2 = 1,jn
         if (jd(i2)-1) 20,20,19
19       jq =jq+1
           goto 10
20     continue
21   return
         end
```